



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/630,705	07/31/2003	Taketo Heishi	2003_1076A	1590
513	7590	09/11/2006	EXAMINER	
WENDEROTH, LIND & PONACK, L.L.P. 2033 K STREET N. W. SUITE 800 WASHINGTON, DC 20006-1021				TECKLU, ISAAC TUKU
ART UNIT		PAPER NUMBER		
		2192		

DATE MAILED: 09/11/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/630,705	HEISHI ET AL.
	Examiner Isaac T. Tecklu	Art Unit 2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) Responsive to communication(s) filed on 31 July 2003.

2a) This action is FINAL. 2b) This action is non-final.

3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) Claim(s) 1-41 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5) Claim(s) _____ is/are allowed.

6) Claim(s) 1-41 is/are rejected.

7) Claim(s) _____ is/are objected to.

8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) The specification is objected to by the Examiner.

10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a) All b) Some * c) None of:

1. Certified copies of the priority documents have been received.
2. Certified copies of the priority documents have been received in Application No. _____.
3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date. _____
3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date <u>05/20/05, 02/28/06</u> .	5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)
	6) <input type="checkbox"/> Other: _____

DETAILED ACTION

1. This action is responsive to the application filed on 07/31/2006.
2. Claims 1- 41 have been examined.

Oath/Declaration

3. The office acknowledges receipt of a properly signed oath/declaration filed on 07/31/2006.

Claim Objections

4. Claims 5 and 6, recite acronym “SIMD”, such acronym should be spelled out once in the claims as its intended meaning and utility are likely will be changed over time. Appropriate correction is required.
5. Claim 14 recites, “the function is counts the number of bits” at lines 5-6. The limitation in the claims are ambiguous because it is not clear whether the function is counts the number of bits represented as a series of the same value as the most significant bit from the next bit to the most significant bit. Appropriate correction is required.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.
7. Claims 9-27 and 38-39 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.
8. Claim 9 recites the limitation "the function" in line 4. There is insufficient antecedent basis for this limitation in the claim.

Claims 10-27 are rejected for dependency upon rejected claim 9.

Claim 38 recites the limitation "the compiler" in line 3. There is insufficient antecedent basis for this limitation in the claim.

Claim 39 recites the limitation "the compiler" in line 10. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 101

9. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

10. Claims 1-36, 37-39 and 40 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1 and 40 are non-statutory as being "A compiler" without being supported by hardware such as tangible computer storage or execution engine, which would enable one skill in the art to construe that the compiler is built from tangible product to carry out any functionality being conveyed from the claim. Thus, the compiler is software *per se* and therefore is not being tangibly embodied in a manner as to be executable.

Under the Interim Guidelines Section IV (a) computer programs claimed as computer listings *per se*, i.e., the descriptions or expressions of the programs are not physical "things." They are neither computer components nor statutory process, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized.

Claims 2-36 are rejected for failing to cure the deficiencies of the above rejected non-statutory claim 1 above.

Claims 37-39 are non-statutory because the claims recite a data structure having non-functional descriptive elements. As such, the claims do not provide any action or interaction between the recited structural elements in order to enable a reasonable interpretation that a concrete, tangible, and useful result is present or yielded based on such interaction or actions taken.

Under the Interim Guidelines Section IV (b) non functional descriptive elements that do not constitute a statutory process, machine, manufacture or composition of matter are non statutory.

Claim Rejections - 35 USC § 102

11. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

12. Claims 1, 28-36 and 40-41 are rejected under 35 U.S.C. 102(b) as being anticipated by Inoue (US 5,758,164).

As per claim 1 Inoue discloses a compiler that translates a source program (e.g. FIG. 1, element 70 and related text) into a machine language program (e.g. FIG.1, element 100 and related text), including operation definition information in which operation that corresponds to a machine language instruction specific to a target processor is defined, comprising:

a parser step of analyzing the source program (e.g. FIG.1, element 20 and related text);

an intermediate code conversion step of converting the analyzed source program into intermediate codes (e.g. FIG. 1, element 80 and related text);

an optimization step of optimizing the converted intermediate codes (e.g. FIG. 1, element 90 and related text); and

a code generation step of converting the optimized intermediate codes into machine language instructions, wherein the intermediate code conversion step includes (e.g. FIG. 1, element 60 and related text):

a detection sub-step of detecting whether or not any of the intermediate codes refer to the operation defined in the operation definition information (in column 17, lines 50-65 "... when the line of intermediate code as checked at step 807, FIG. 8A ..."); and

a substitution sub-step of substituting the intermediate code with a corresponding machine language instruction, when the intermediate code is detected (in column 17, lines 60-65 "... check is performed to whether ..."), and in the optimization step, the intermediate codes are optimized (in column 9, lines 5-10 the intermediate code of the table 4... judgment is made ... intermediate code is read out ..." and e.g. FIG. 3A and related text), the intermediate codes including the machine language instruction substituted in the substitution sub-step (in column 17, lines 60-5 "... value of the variable is varied the variable ...").

As per claim 28, Inoue discloses the compiler according to claim 1, wherein the optimization step includes a type conversion sub-step of substituting a plurality of intermediate codes or machine language instructions (e.g. FIG.1, element 60 and related text) that perform an operation between different types with one machine language instruction that performs said operation (in column 6, lines 40-51 "... the code generating portion 60 of FIG. 1 converts the optimized intermediate code ...").

As per claim 29, Inoue discloses the compiler according to claim 28, wherein in the type conversion sub-step, a plurality of intermediate codes or machine language instructions that perform an operation that multiplies two n-bit variants and stores the result in a 2n-bit variant are substituted with one machine language instruction that performs said operation (in column 13, lines 65-67 and in column 14, lines 5-10 "... instruction of code C: =A ADD B").

As per claim 30, Inoue discloses the compiler according to claim 29, wherein in the type conversion sub-step, the operation is substituted with the machine language instruction when an explicit declaration that a type conversion from n bits to 2n bits is carried out is made toward the two variants (e.g. FIG. 4 and related text).

As per claim 31, Inoue discloses the compiler according to claim 1, wherein the compiler targets a processor that has two or more fixed point modes of performing an operation targeting two or more fixed point types (e.g. TABLE 1, B, C and related text), in the parser step, a description that the fixed point mode is switched is detected in the source program (e.g. FIG.1, element 20 and related text), and the compiler further includes a fixed point mode switch step of inserting a machine language instruction to switch fixed point modes following the description that the fixed point mode is switched, when the description is detected in the parser step (e.g. FIG. 8A, element 801 and 802 and related text).

As per claim 32, Inoue discloses the compiler according to claim 31, wherein the description that the fixed point mode is switched is associated with a target function, and in the fixed point mode switch step, machine language instructions for saving and returning of the fixed point mode are inserted respectively into the head and the tail of a corresponding function (e.g. FIG. 8A, element 801 and 802 and related text).

As per claim 33, Inoue discloses the compiler according to claim 1, wherein the optimization step includes a latency optimization sub-step of detecting a description in the source program, the description designating latency in which execution time at the specific position is secured only for a predetermined number of cycles, and scheduling a machine language instruction so that the latency is secured according to the detected designation (e.g. FIG. 1, element 90 and related text).

As per claim 34, Inoue discloses the compiler according to claim 33, wherein in the latency optimization sub-step, when a description that latency of a predetermined cycles is designated targeting an interval between a first machine language instruction to which a first

label is attached and a second machine language instruction to which a second label is attached is detected (e.g. FIG. 1, element 90 and related text), the scheduling is executed in order that it takes execution time of only the number of cycles since the first machine language instruction is executed until the second machine language instruction is executed (in column 17, lines 60-5 “... value of the variable is varied the variable ...”).

As per claim 35, Inoue discloses the compiler according to claim 33, wherein in the latency optimization sub-step, when a description that latency of a predetermined cycles is designated targeting access to a specified register is detected (in column 17, lines 60-5 “... value of the variable is varied the variable ...”), the scheduling is executed in order that it takes execution time of only the number of cycles since a machine language instruction to access the register is executed until a machine language instruction to access said register is executed next time (e.g. FIG. 4 and related text).

As per claim 36, Inoue discloses the compiler according to claim 1, wherein the compiler further comprises a class library to substitute a machine language instruction used in the operation definition information with a machine language instruction of a second processor that is different from a first processor that said compiler targets (in column 17, lines 60-5 “... value of the variable is varied the variable ...”).

As per claim 40, this is the apparatus version of the claimed compiler discussed above (Claim 1), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, this claim is also anticipated by Inoue.

As per claim 40, this is the apparatus version of the claimed method discussed above (Claim 1), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, this claim is also anticipated by Inoue.

13. Claims 37-39 are rejected under 35 U.S.C. 102(b) as being anticipated by Sterling et al. (US 5,822,588).

As per claim 37, Sterling discloses a computer-readable recoding medium on which a header file included in a source program to be compiled is recorded (e.g. FIG. 4B element 102, element 96 and related text), wherein operation definition information in which operation that corresponds to a machine language instruction specific to a target processor is defined is a header file included in the source program (e.g. FIG. 5, element 124-130 and related text), in the header file, the operation is defined by a class made up of data and a method (e.g. FIG. 6 and related text).

As per claim 38, Sterling discloses a computer-readable recoding medium on which a class library included in a source program to be compiled is recorded (e.g. FIG. 4B element 102, element 96 and related text), wherein the compiler further comprises a class library to substitute a machine language instruction used in the operation definition information (e.g. FIG. 4B, element 104 and related text) with a machine language instruction of a second processor that is different from a first processor that said compiler targets (e.g. FIG. 4B, element 96 and related text).

As per claim 39, Sterling discloses a computer-readable recording medium on which a source program to be compiled including at least one of a header file or a class library is recorded (e.g. FIG. 4B element 102, element 96 and related text), wherein operation definition information in which operation that corresponds to a machine language instruction specific to a target processor is defined is a header file included in the source program (e.g. FIG. 5, element 124-130 and related text), in the header file, the operation is defined by a class made up of data and a method, and the compiler further comprises a class library to substitute a machine language instruction used in the operation definition information with a machine language instruction of a second processor that is different from a first processor that said compiler targets (e.g. FIG. 4B, element 96 and related text).

Claim Rejections - 35 USC § 103

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

15. Claims 2-4, 7-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inoue (US 5,758,164) in view of Sterling et al. (US 5,822,588).

As per claim 2, Inoue discloses the operation is defined by a class made up of data and a method (in column 1, lines 45-50 TABLE 1, A = B op C),

Inoue does not explicitly disclose the compiler according to claim 1, wherein the operation definition information is a header file to be included in the source program, in the intermediate code conversion step, whether or not any of the intermediate codes refer to the operation is detected by detecting whether or not any of the intermediate codes refer to the class defined by the header file. However Sterling in an analogous art discloses the compiler according to claim 1, wherein the operation definition information is a header file to be included in the source program (e.g. FIG. 5, element 124-130 and related text and in column 15, lines 1-29 #include <note.h>, #include <assrt.h> and related text), in the intermediate code conversion step, whether or not any of the intermediate codes refer to the operation is detected by detecting whether or not any of the intermediate codes refer to the class defined by the header file (in column 8, lines 51-60 "... syntactic and semantic analysis ..." e.g. FIG. 4B, element 88 and related text). Therefore it would have been obvious to one skilled in the art at the time of the invention was made to include in the source program disclosed by Inoue the header file to provide the ability to alter the interpretation of system for making interposing possible without creating a maintenance burden as once suggested by Sterling (in column 10, lines 60-68).

As per claim 3, Inoue discloses the compiler according to claim 2, wherein the class defines a fixed point type (e.g. TABLE 1, B, C and related text), and in the detection sub-step, intermediate codes that use the fixed point type data are detected (in column 9, lines 5-15 "...

intermediate code of the table 4, the code “OPTBEG” is read at first ... judgment is made ...” and in column 10, lines 5-25 TABLE 5 “OPTBEG”).

As per claim 4, Inoue discloses the compiler according to claim 3, wherein the method in the class defines operators targeting the fixed point type data (e.g. TABLE 1, A, B and related text),

in the detection sub-step, the detection is executed based on whether or not a set of the operator and the data type targeting an operation agrees with the definition in the method (e.g. FIG. 8A, element 801 and 802 and related text), and

in the substitution step, an intermediate code whose set of the operator and the data type agrees with the definition is substituted with a corresponding machine language instruction (e.g. FIG. 15, element 1505 and related text).

As per claim 7, Inoue discloses the compiler according to claim 2, wherein the class is associated with one machine language instruction that realizes the corresponding processing, and in the substitution sub-step, the intermediate code is substituted with one machine language instruction associated with the class (e.g. FIG. 15, element 1505 and related text).

As per claim 8, Inoue discloses the compiler according to claim 2, wherein the class is associated with two or more machine language instructions that realize the corresponding processing (e.g. FIG. 15, element 1501 and related text), and in the substitution sub-step, the intermediate codes are substituted with two or more machine language instructions associated with the class (e.g. FIG. 15, element 1505 and related text).

As per claim 9, Inoue discloses, the operation is defined by the function (in column 2, lines 5-20 TABLE 2 and related text)

Inoue does not explicitly disclose the compiler according to claim 1, wherein the operation definition information is a header file to be included in the source program, in the header file, the operation is defined by the function, and in the intermediate code conversion step, whether or not any of the intermediate codes refer to the operation is detected by detecting

whether or not any of the intermediate codes refer to the function defined by the header file. However Sterling in an analogous art discloses the compiler according to claim 1, wherein the operation definition information is a header file to be included in the source program (e.g. FIG. 5, element 124-130 and related text and in column 15, lines 1-29 #include <note.h>, #include <assrt.h> and related text), in the intermediate code conversion step, whether or not any of the intermediate codes refer to the operation is detected by detecting whether or not any of the intermediate codes refer to the class defined by the header file (in column 8, lines 51-60 "... syntactic and semantic analysis ..." e.g. FIG. 4B, element 88 and related text). Therefore it would have been obvious to one skilled in the art at the time of the invention was made to include in the source program disclosed by Inoue the header file to provide the ability to alter the interpretation of system for making interposing possible without creating a maintenance burden as once suggested by Sterling (in column 10, lines 60-68).

As per claim 10, Inoue discloses the compiler according to claim 9, wherein the function describes one machine language instruction that realizes the corresponding processing, and in the substitution sub-step, the intermediate code is substituted with one machine language instruction described in the function (e.g. FIG. 1, element 60 and related text).

As per claim 11, Inoue discloses the compiler according to claim 10, wherein the function includes a function that returns the number of bits represented as a series of Os from the most significant bit of input data (in column 15, lines 35-40 "... value of variable ..."), and

a machine language instruction described in the function counts the number of bits represented as a series of Os from the most significant bit of a value stored in the first register and stores the result in the second register (in column 15, lines 35-41 "... variation of the value is stored in the base block...").

As per claim 12, Inoue discloses the compiler according to claim 10, wherein the function includes a function that returns the number of bits represented as a series of is from the most significant bit (TABLE 15 and related text), and

a machine language instruction described in the function counts the number of bits represented as a series of 1s from the most significant bit concerning the value stored in the first register and stores the result in the second register (in column 15, lines 35-41 "... variation of the value is stored in the base block...").

As per claim 13, Inoue discloses the compiler according to claim 10, wherein the function includes a function that returns the number of bits that the same value as the most significant value of input data succeeds (TABLE 15 and related text), and

a machine language instruction described in the function counts the number of bits represented by a series of the same value as the most significant bit of the value stored in the first register and stores the result in the second register (in column 15, lines 35-41 "... variation of the value is stored in the base block...").

As per claim 14, Inoue discloses the compiler according to claim 13, wherein the function returns the number of bits represented as a series of the same value as the most significant value of input data from the next bit to the most significant bit (TABLE 15 and related text), and

a machine language instruction described in the function counts the number of bits represented as a series of the same value as the most significant bit from the next bit to the most significant bit of the value stored in the first register and stores the result in the second register (in column 15, lines 35-41 "... variation of the value is stored in the base block...").

As per claim 15, Inoue discloses the compiler according to claim 10, wherein the function includes a function that returns the number of bits of 1 included in input data, and

a machine language instruction described in the function counts the number of bits of 1 of the value stored in the first register and stores the result in the second register (in column 15, lines 35-41 "... variation of the value is stored in the base block..." FIG. 4, BASE BLOCK 2).

As per claim 16, Inoue discloses the compiler according to claim 10, wherein the function includes a function that returns an sign-expanded value based on bits extracted at designated bit positions from input data (TABLE 15 and related text), and

a machine language instruction described in the function takes out bits at the bit positions designated by the second register from the value stored in the first register, sign-expands said bits and stores the sign-expanded bits in the third register (e.g. FIG. 4, BASE BLOCK 3).

As per claim 17, Inoue discloses the compiler according to claim 10, wherein the function includes a function that returns an zero-expanded value based on bits extracted at designated bit positions from input data (TABLE 15 and related text), and

a machine language instruction described in the function takes out bits at the bit positions designated by the second register from the value stored in the first register, zero-expands said bits and stores the zero-expanded bits in the third register (e.g. FIG. 4, BASE BLOCK 3).

As per claim 18, Inoue discloses the compiler according to claim 9, wherein the function describes a machine language instruction sequence including two or more machine language instructions that realize corresponding processing, and

in the substitution sub-step, the intermediate codes are substituted with the machine language instruction sequence (e.g. FIG. 1, element 100 and related text).

As per claim 19, Inoue discloses the compiler according to claim 18, wherein the function includes a function that updates an address of modulo addressing (e.g. Fig. 1, element 90 and related text).

As per claim 20, Inoue discloses the compiler according to claim 19, wherein the machine language instruction sequence described in the function includes a machine language instruction that stores in the third register a value acquired by substituting a predetermined bit field of the value stored in the first register with a value stored in the second register (e.g. FIG. 4, and related text).

As per claim 21, Inoue discloses the compiler according to claim 18, wherein the function includes a function that updates an address of bit reverse addressing (e.g. FIG. 8A, element 802 and related text).

As per claim 22, Inoue discloses the compiler according to claim 21, wherein the machine language instruction sequence described in the function includes a machine language instruction that stores in the third register a value acquired by inverting bit by bit a position of a predetermined bit field of the value stored in the first register (e.g. FIG. 4, and related text).

As per claim 23, Inoue discloses the compiler according to claim 9, wherein the function includes a function that can designate a temporary variable with the accumulator as a reference type, the function being an operation of updating both an accumulator that does not target allocation in optimization and a general purpose register that targets allocation in optimization (e.g. FIG. 1, element 40 and related text).

As per claim 24, Inoue discloses the compiler according to claim 23, wherein the function performs a multiplication and can designate a temporary variable with an accumulator as a reference type, the accumulator storing the result of the multiplication (TABLE 2 and related text).

As per claim 25, Inoue discloses the compiler according to claim 23, wherein the function performs a sum of products (TABLE 12 and related text) and can designate a temporary variable with an accumulator as a reference type, the accumulator storing the result of the sum of products (TABLE 16 and related text).

As per claim 26, Inoue discloses the compiler according to claim 9, wherein in the substitution sub-step, the intermediate code referring to the function is substituted with a machine language instruction having a variety of operands corresponding to a variety of arguments of said function (TABLE 16 and related text).

As per claim 27, Inoue discloses the compiler according to claim 26, wherein in the substitution sub-step, an intermediate code referring to the function is substituted with (i) a machine language instruction whose operand is a constant value acquired by holding in constants when all arguments are constants (TABLE 16 and related text); (ii) a machine language instruction that has an immediate value operand when a part of arguments are

constants (TABLE 14 and related text); and (iii) a machine language instruction that has a register operand when all arguments are variable (e.g. FIG. 4 and related text).

16. Claims 5-6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inoue (US 5,758,164) in view of Bik et al. (US 2001/0006667 A1).

As per claim 5, Inoue does not explicitly disclose the compiler according to claim 2, wherein the class defines a SIMD type, and in the detection sub-step, the intermediate code using the SIMD type data is detected. However Bik discloses SIMD instruction statements which are represented in the intermediate code form utilized within an internal representation of the source program code. Once the codes are generated the system compiler replaces the one or more identified code with the generated code (in paragraph [0102]). Therefore it would have been obvious to one skilled in the art at the time of the invention was made to use the SIMD type data enable simultaneous or parallel processing of instruction on multiple data element (in paragraph [0004]) and to provide significant enhancement to execution to execution bandwidth (in paragraph [0004]).

As per claim 6, Inoue discloses in the detection sub-step, the detection is executed based on whether or not a set of the operator and the data type targeting an operation agrees with the definition in the method (e.g. FIG. 8A, element 801 and 802 and related text), and

in the substitution step, an intermediate code whose set of the operator and the data type agrees with the definition is substituted with a corresponding machine language instruction (e.g. FIG. 15, element 1505 and related text).

Inoue does not explicitly disclose class defined the operator targeting the SIMD type data. However Bik discloses SIMD instruction statements which are represented in the intermediate code form utilized within an internal representation of the source program code. Once the codes are generated the system compiler replaces the one or more identified code with the generated code (in paragraph [0102]). Therefore it would have been obvious to one skilled in the art at the time of the invention was made to use the SIMD type data enable simultaneous

Art Unit: 2192

or parallel processing of instruction on multiple data element (in paragraph [0004]) and to provide significant enhancement to execution to execution bandwidth (in paragraph [0004]).

Conclusion

17. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Isaac T. Tecklu whose telephone number is (571) 272-7957. The examiner can normally be reached on M-TH 9:300A - 8:00P.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Isaac Tecklu

Art Unit 2192



TUAN DAM
SUPERVISORY PATENT EXAMINER